```java
package final1alejandroCastellanos;

import java.io.Serial;
import java.io.Serializable;
import java.util.Objects;

public class Asignatura implements Serializable {

    @Serial
    private static final long serialVersionUID = 2223L;

    public final String nombre;
    public final int codigo;

    //Usamos atributo estatico para mantener la cuenta
    private static int contadorCodigo = 1;

    public Asignatura(String nombre) {
        this.nombre = nombre;
        codigo = contadorCodigo++;
    }

    /**
     * Equals definiendo que 2 Asignaturas son iguales si tienen el mismo nombre
     * @param o Objeto al que se compara
     * @return True si el objeto es de clase 'Asignatura' y tiene el mismo nombre
     */
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Asignatura that = (Asignatura) o;
        return Objects.equals(nombre, that.nombre);
    }

    @Override
    public int hashCode() {
        return Objects.hash(nombre);
    }
```

```
40
41      @Override
42      public String toString() {
43          return nombre;
44      }
45  }
46
```

```java
package final1alejandroCastellanos;

import java.io.Serial;
import java.io.Serializable;
import java.util.*;
import java.util.concurrent.ThreadLocalRandom;

public class Estudiante implements Serializable {

    @Serial
    private static final long serialVersionUID = 2223L;

    private final String identificador;
    private String nombre;
    private String apellidos;
    private String correo;
    private Map<Asignatura, Double> asignaturas;

    public Estudiante(String identificador, String nombre, String apellidos, String correo) {
        this.identificador = identificador;
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.correo = correo;
        this.asignaturas = new HashMap<>();
    }

    /**
     * Generamos un pin de una longitud 1-10 formado por numeros no repetidos
     * @param num Longitud del pin
     * @return pin guardado en tipo Long
     */
    public long obtenerPin(int num) {
        if (num > 10 || num < 0) num = 10;

        StringBuilder numerosLibres = new StringBuilder("0123456789");
        StringBuilder cadenaPin = new StringBuilder();

        // Vamos añadiendo numeros a una cadena nueva segun un numero aleatorio que determina la posicion en la cadena
        //de numeros posibles.
```

```
40          //  Luego eliminamos el numero de dicha cadena para evitar repeticiones
41          for (int i = 0; i < num; i++) {
42              int posicionNum = ThreadLocalRandom.current().nextInt(0, numerosLibres.length());
43              cadenaPin.append(numerosLibres.charAt(posicionNum));
44              numerosLibres.deleteCharAt(posicionNum);
45          }
46
47          return Long.parseLong(cadenaPin.toString());
48      }
49
50      /**
51       * Generamos un pin de longitud 4 formado por numeros no repetidos
52       * @return pin guardado en tipo Long
53       */
54      public long obtenerPin(){
55          return obtenerPin(4);
56      }
57
58      /**
59       * Genera un login formado por los 3 primeros caracteres del nombre + 3 primeros del apellido.
60       * De no ser posible, se toman los caracteres que se pueda
61       * @return pin guardado en tipo Long
62       */
63      public String obtenerLogin(){
64          String login;
65          //Separamos para solo obtener el primer apelliddo
66          String[] apellidosArray = apellidos.split(" ");
67          String primerApellido = apellidosArray[1];
68
69          if(nombre.length()<3) login = nombre;
70          else login = nombre.substring(0,3);
71
72          if(primerApellido.length()<3) login += primerApellido;
73          else login += primerApellido.substring(0,3);
74
75          return login.toLowerCase();
76      }
77
78      /**
```

```java
     * Comprueba si el atributo correo tiene formato valido
     * @return True si el correo esta bien formateado
     */
    public boolean esCorreoValido(){
        return correo.matches("^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\\.[a-zA-Z0-9-]+)*$");
    }

    /**
     * Añade a un alumno al Mapa de Asignatura-Notas (llamado asignaturas) a partir de un objeto Asignatura
     * @param asignatura Asignatura en al que se matricula (crea entrada en el mapa)
     */
    public void matricular(Asignatura asignatura){
        asignaturas.put(asignatura,0.0);
    }

    /**
     * Cambia la nota almacenada en el mapa de asignaturas a partir de un codigo de asignatura
     * @param codigoBuscado Codigo de la asignatura a cambiar
     * @param nota Nota nueva
     * @throws Exception Lanzada si no existe una Asignatura con ese codigo en el mapa
     */
    public void cambiarNota(int codigoBuscado, double nota) throws Exception {
        //Recorremos las asignaturas guardadas hasta encontrar una que tenga el codigo que buscamos
        for (Asignatura asignatura : asignaturas.keySet()) {
            if(asignatura.codigo==codigoBuscado){
                asignaturas.put(asignatura,nota);
                return;
            }
        }

        //Si no encontramos una asignatura, lanzamos error con este mensaje
        throw new Exception("No se ha encontrado una asignatura con ese codigo");
    }

    /**
     *  Cambia la nota almacenada en el mapa de asignaturas a partir de un nombre de asignatura
     * @param nombreBuscado Nombre de la asignatura a cambiar
     * @param nota Nota nueva
     * @throws Exception Lanzada si no existe una Asignatura con ese codigo en el mapa
```

```
118      */
119     public void cambiarNota(String nombreBuscado, double nota) throws Exception {
120         //Igual que el anterior metodo pero a partir del nombre
121         for (Asignatura asignatura : asignaturas.keySet()) {
122             if(asignatura.nombre.equals(nombreBuscado)){
123                 asignaturas.put(asignatura,nota);
124                 return;
125             }
126         }
127 
128         throw new Exception("No se ha encontrado una asignatura con ese codigo");
129     }
130 
131     /**
132      * Comprueba si un alumno promociona o no
133      * @return True si nota>5 en todas las entradas del mapa
134      */
135     public boolean promociona(){
136         for (Double nota : asignaturas.values()) {
137             if(nota<5)
138                 return false;
139         }
140         return true;
141     }
142 
143     //TODO: Usar stream
144     /**
145      * Muestra todas las entradas del mapa 'asignaturas' del estudiante
146      */
147     public void mostrarNotas(){
148         ArrayList<Map.Entry<Asignatura,Double>> listAsignaturas = new ArrayList<>(asignaturas.entrySet());
149 
150         listAsignaturas.sort(new Comparator<Map.Entry<Asignatura, Double>>() {
151             @Override
152             public int compare(Map.Entry<Asignatura, Double> o1, Map.Entry<Asignatura, Double> o2) {
153                 return o1.getKey().nombre.compareTo(o2.getKey().nombre);
154             }
155         });
156 
```

```
157         //Puede ser simplificado a esto, ya que podemos obtener la clave de comparacion
158         //listAsignaturas.sort(Comparator.comparing(entry -> entry.getKey().nombre));
159 
160         System.out.printf("%10s:%-35s::%4s",
161                 "Codigo", "Nombre de Asignatura", "Nota"
162         );
163 
164         for (Map.Entry<Asignatura, Double> entry : listAsignaturas) {
165             System.out.printf("%n%10d:%-35s::%.2f",
166                     entry.getKey().codigo, entry.getKey().nombre, entry.getValue()
167             );
168         }
169     }
170 
171     public Map<Asignatura, Double> getAsignaturas() {
172         return asignaturas;
173     }
174 
175     @Override
176     public String toString() {
177         return String.format("%-15s - %-15s - %-25s - %-30s",
178                 identificador, nombre, apellidos, correo
179         );
180     }
181 }
182 
```

```java
package final1alejandroCastellanos;

import java.io.*;
import java.util.ArrayList;
import java.util.Map;


public class LectorFicheroEstudiantes {

    public static void main(String[] args) {

        // Pruebas sin archivo
//        ArrayList<Estudiante> listaPruebas = new ArrayList<>();
//        Estudiante ale = new Estudiante("ABC","Alex","Caste Dalm","alex@caste.com");
//        Estudiante ale2 = new Estudiante("ABC2","Alex2","Caste Dalm","alex@caste.com");
//        listaPruebas.add(ale);
//        listaPruebas.add(ale2);
//        ale.matricular(new Asignatura("Mates"));
//        ale2.matricular(new Asignatura("LENG"));
//
//        try {
//            ale.cambiarNota("Mates",6.4);
//            ale.cambiarNota("Mates",6.4);
//        } catch (Exception e) {
//            System.out.println(e.getMessage());
//        }
//            for (Estudiante estudiante : listaPruebas) {
//                if(estudiante.promociona()) listaPromocionados.add(estudiante);
//                else listaNoPromocionados.add(estudiante);
//            }

        File archivo = new File("Resources/estudiantes.dat");

        try(FileInputStream fis = new FileInputStream(archivo);
            BufferedInputStream bis = new BufferedInputStream(fis);
            ObjectInputStream ois = new ObjectInputStream(bis))
        {

            ArrayList<Estudiante> listaPromocionados = new ArrayList<>();
```

```
40              ArrayList<Estudiante> listaNoPromocionados = new ArrayList<>();
41
42              while(bis.available()>0){
43                  Estudiante estudiante = (Estudiante) ois.readObject();
44                  if(estudiante.promociona()) listaPromocionados.add(estudiante);
45                  else listaNoPromocionados.add(estudiante);
46              }
47
48
49              // Estudiantes promocionados
50              System.out.println("==========================");
51              System.out.println("Estudiantes promocionados: ");
52              System.out.println("---------------------------");
53              System.out.printf("%-15s - %-15s - %-25s - %-30s %n",
54                      "Identificador", "Nombre", "Apellidos", "Correo"
55              );
56
57              for (Estudiante estudiante : listaPromocionados) {
58                  System.out.println(estudiante);
59                  System.out.println("--- Notas ---");
60                  estudiante.mostrarNotas();
61
62                  Double sumaNotas = 0.0;
63                  for (Double nota : estudiante.getAsignaturas().values()) {
64                      sumaNotas+=nota;
65                  }
66                  System.out.println("\nMedia: "+sumaNotas/estudiante.getAsignaturas().size());
67              }
68
69
70              // Estudiantes NO promocionados
71              System.out.println("==========================");
72              System.out.println("Estudiantes NO promocionados: ");
73              System.out.println("-----------------------------");
74              System.out.printf("%-15s - %-15s - %-25s - %-30s %n",
75                      "Identificador", "Nombre", "Apellidos", "Correo"
76              );
77              for (Estudiante estudiante : listaNoPromocionados) {
78                  System.out.println(estudiante);
```

```
79                  System.out.println("--- Notas ---");
80                  estudiante.mostrarNotas();
81
82                  System.out.println("\nAsignaturas suspendidas:");
83                  for (Map.Entry<Asignatura, Double> entry : estudiante.getAsignaturas().entrySet()) {
84                      if(entry.getValue()<5)
85                          System.out.println(entry);
86                  }
87              }
88
89
90          } catch (FileNotFoundException e) {
91              System.out.println("Archivo no encontrado");
92          } catch (IOException e) {
93              e.printStackTrace();
94          } catch (ClassNotFoundException e) {
95              e.printStackTrace();
96          }
97      }
98  }
99
```